WHAT MAKES FOR A HIT POP SONG?

NICK BORG AND GEORGE HOKKANEN

Overview

The possibility of a hit song prediction algorithm is both academically interesting and industry motivated. While several companies currently attest to their ability to make such predictions, publicly available research (most notably Pachet, [1]) suggests that current methods are unlikely to produce accurate predictions. We are currently investigating to what extent features extracted from the music itself can be used to classify popular music. In order to address this question properly, we believe that we must also investigate the social factors that dictate music popularity. One of these factors, discussed in [19] is preferential attachment, the idea that once an artist becomes popular enough it is more likely that the number of plays of that artist's tracks becomes skewed in their favor despite the fact that the artist's new songs may depart substantially from the music that first made the artist popular. Depending on the success of this feature/popularity correlation, we may be motivated after the milestone to attempt correlations of popularity (via billboard charts and youtube views) with twitter data, but this is a consideration for the future.

Dataset

Our analyses to date have been centered around the million song dataset subset which consists of ten thousand songs with metadata, chroma, and mfcc-like coefficients. The dataset is distributed by Labrosa for free use [3]. One of the features given in the dataset is a popularity value called 'hotttnesss' produced by the Echonest, but this value appears too sparse in the data to be useful. For a popularity replacement measure, we have created a dataset of youtube view counts (coordinated with the subset only at the moment), ratings, and average rating of the first result for each query of artist name, song name. We checked by hand the accuracy of this scraping method and concluded that the two errors in thirty randomly drawn songs was not a problem given that the errors also coordinate well with very low view counts (i.e. something unpopular enough to not return a copy of the song on youtube ends up returning an unrelated video with a low view count).

Preliminary Analysis

In order to familiarize ourselves with the data, we first ran basic correlation coefficients between different parts of the metadata and also with our extracted youtube view counts. The results were largely insignificant and included weak correlations such as one of .2 between the tempo and loudness metadata features. Correlations between the youtube view counts and the echonest metadata features loudness, tempo, hotttness, and danceability were completely negligible (less than .05 in magnitude). The fact that these are not at all correlated is interesting in its own right because it points to no single metadata feature being at all a good predictor of views on youtube.

We then made a shot in the dark, taking some number of mfcc's (usually less than 100) and concatenating them into a vector of length less than 1200. We trained an SVM using these vectors to predict the youtube view counts obtained through scraping, using several different kernels all to no avail (the accuracy was always less than .6). Intuitively, mfcc's provide a rather detailed description of a song over time. One way to account for the poor performance of our SVMs is by noting that treating the mfcc's as a vector of unordered features fails to account for the temporal progression of a song. For this reason, we hypothesized that a more intelligent implementation of an SVM trained on mfccs would use a string kernel, providing our classifier with a symbolic representation of the song over time. Our efforts in this direction are outlined in the following sections on Feature Extraction and Classification.

Feature Extraction

Given that the mfcc and spectral data is temporal, we wanted to use the ordering therein to describe the sound. Motivating a string kernel SVM approach, we creating 'string' features for our songs as follows.

For each i, we take the spectral bucket i (corresponding to a frequency-aggregate magnitude) for each spectra vector within a range of each song (usually about 45 seconds in the middle). This gives a list of values which correspond to the magnitudes over time of this slice of the sound spectrum for every song.

Next, using a subset of this data (we used two hundred of the ten thousand songs) we compute a list of intervals (we used 26 of them, corresponding to the characters a through z) that uniformly distribute the data. Then using these intervals, we compute a string for each sequence of data obtained in the first step by replacing each value with a symbol or letter that represents the interval.

In the following, we outline the implementation of our classifier.

Classification

We chose to implement our classifier using the freely available libsvm string extension [4],[5]. SVM on xyz. Naive bayes on tags.

Results

Acknowledgements

[1] Pachet, F. and Roy, P. (2008) Hit Song Science is Not Yet a Science. Proceedings of Ismir 2008, pages 355-360, Philadelphia, USA

[2] Salganik, M. J. Dodds, P. S. Watts, D. J. Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market, Science, 311, 854-856, 2006

[3] Million Song Dataset, official website by Thierry Bertin-Mahieux, available at: http://labrosa.ee.columbia.edu/millionsong/

[4] LIBSVM citation to go here [5] LIBSVM-String citation here.